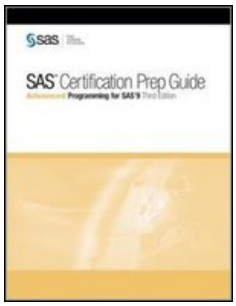


SAS Certification Prep Guide: Advanced Programming for SAS 9, Third Edition

by SAS Institute

Chapter 17: Formatting Data

Overview

Introduction

Custom formats are used to display variable values in certain ways, such as formatting a product number so that it is displayed as descriptive text. You should already be familiar with using the FORMAT procedure to create and store formats.

In this chapter you learn to document formats, use formats located in any catalog, create formats with overlapping ranges, and use the PICTURE statement to create a format for printing numbers. You also learn an easy way to update formats by creating a SAS data set from a format, updating the data set, and then creating a format from the SAS data set.

Objectives

In this chapter, you learn to

- create formats with overlapping ranges
- create custom formats using the PICTURE statement
- use the FMTLIB keyword with the FORMAT procedure to document formats
- use the CATALOG procedure to manage format entries
- use the DATASETS procedure to associate a format with a variable
- use formats located in any catalog
- substitute formats to avoid errors
- create a format from a SAS data set
- create a SAS data set from a custom format.

Creating Custom Formats Using the VALUE Statement

Review of Creating Non-Overlapping Formats

As you learned in *Creating and Applying User-Defined Formats*, you can use the VALUE statement in the FORMAT procedure to create a custom format for displaying data in a particular way. For example, suppose you have airline data and you ant to create several custom formats that you can use for your report-riting tasks. You need formats that

- group airline routes into zones
- label airport codes as International or Domestic
- group cargo revenue figures into ranges.

The following PROC FORMAT step creates these three formats:

```
libname library 'c:\sas\newfmts';
proc format lib=library;
   value $routes
         'Route1' = 'Zone 1'
```

```
          'Route2' - 'Route4' = 'Zone 2'
          'Route5' - 'Route7' = 'Zone 3'
          ' ' = 'Missing'
          other = 'Unknown';
   value $dest
          'AKL','AMS','ARN','ATH','BKK','BRU',
          'CBR','CCU','CDG','CPH','CPT','DEL',
          'DXB','FBU','FCO','FRA','GLA','GVA',
          'HEL','HKG','HND','JED','JNB','JRS',
          'LHR','LIS','MAD','NBO','PEK','PRG',
          'SIN','SYD','VIE','WLG' = 'International'
          'ANC','BHM','BNA','BOS','DFW','HNL',
          'IAD','IND','JFK','LAX','MCI','MIA',
          'MSY','ORD','PWM','RDU','SEA','SFO' = 'Domestic';
   value revfmt
          . = 'Missing'
          low - 10000 = 'Up to $10,000'
          10000 <- 20000 = '$10,000+ to $20,000'
          20000 <- 30000 = '$20,000+ to $30,000'
          30000 <- 40000 = '$30,000+ to $40,000'
          40000 <- 50000 = '$40,000+ to $50,000'
          50000 <- 60000 = '$50,000+ to $60,000'
          60000 <- HIGH  = 'More than $60,000';
run;
```

Tip If you choose to run this example code, be sure to change the path in the LIBNAME statement to a storage location in your operating environment.

The PROC FORMAT step creates three formats: $ROUTES. and $DEST., hich are character formats, and REVFMT., hich is numeric.

$ROUTES. groups airline routes into zones. In $ROUTES.,

- both single values and ranges are assigned labels
- missing values are designated by a space in quotation marks and are assigned the label "Missing"
- the keyword *OTHER* is used to assign the label "Unknon" to any values that are not addressed in the range.

$DEST. labels airport codes as either International or Domestic. In $DEST.,

- unique character values are enclosed in quotation marks and separated by commas
- missing values and values not included in the range are not handled in this format.

REVFMT. groups cargo revenue figures into ranges. In REVFMT.,

- the "less than" operator (<) is used to sho a *non-inclusive range* (`10000<-20000` indicates that the first value is not included in the range)
- the keyword *LOW* is used to specify the loer limit of avariable's value range, but it does not include missing values
- missing values are designated ith a period (.) and assigned the label "Missing"
- the keyword *HIGH* is used to specify the upper limit of a variable's value range.

Creating a Format with Overlapping Ranges

There are times when you need to create a format that groups the same values into different ranges. To create overlapping ranges, use the *MULTILABEL* option in the VALUE statement in PROC FORMAT.

General form, VALUE statement with the MULTILABEL option:

VALUE *format-name* **(MULTILABEL);**

where

format-name

is the name of the character or numeric format that is being created.

Example

Suppose you want to create a format that groups dates into overlapping categories. In the table below, notice that each month appears in two groups.

Value	Label
Jan - Mar	1st Quarter
Apr - Jun	2nd Quarter
Jul - Sep	3rd Quarter
Oct - Dec	4th Quarter
Jan - Jun	First Half of Year
Jul - Dec	Second Half of Year

In the PROC FORMAT step below, the MULTILABEL option has been added to indicate that the DATES, format will have values with overlapping ranges:

```
proc format;
   value dates (multilabel)
         '01jan2000'd - '31mar2000'd = '1st Quarter'
         '01apr2000'd - '30jun2000'd = '2nd Quarter'
         '01jul2000'd - '30sep2000'd = '3rd Quarter'
         '01oct2000'd - '31dec2000'd = '4th Quarter'
         '01jan2000'd - '30jun2000'd = 'First Half of Year'
         '01jul2000'd - '31dec2000'd = 'Second Half of Year';
run;
```

Multilabel formatting allows an observation to be included in multiple rows or categories. To use the multilabel formats, you can specify the MLF option on class variables in procedures that support it:

- PROC TABULATE
- PROC MEANS
- PROC SUMMARY.

The MLF option activates multilabel format processing when a multilabel format is assigned to a class variable. For example, in the following TABULATE procedure,

- the FORMAT= option specifies DOLLAR15.2 as the default format for the value in each table cell
- the FORMAT statement references the new format DATES, for the variable **`Date`**
- the CLASS statement identifies **`Date`** as the class variable and uses the MLF option to activate multilabel format processing
- the row dimension of the TABLE statement creates a row for each formatted value of **`Date`**.

```
proc tabulate data = sasuser.sale2000 format = dollar15.2;
   format Date dates.;
   class Date / mlf;
   var RevCargo;
   table Date, RevCargo*(mean median);
run;
```

	RevCargo	
	Mean	Median

Date		
1st Quarter	$24.839.08	$4.939.00
2nd Quarter	$14.949.77	$4.579.50
3rd Quarter	$19.836.00	$3.591.00
4th Quarter	$20.403.13	$1.940.00
First Half of Year	$19.894.42	$4.823.00
Second Half of Year	$20,261.35	$2.100.00

Tip For more information about using the MULTILABEL option, see the SAS documentation for the FORMAT procedure.

Creating Custom Formats Using the PICTURE Statement

Overview

You have learned that you can use the VALUE statement to convert output values to a different form. Suppose one of the variables in your data set had numeric values that you wanted to format a certain way. For example, you might have a phone number listed in your data set as *1111231234* and you want to format it as *(111) 123-1234*. You can use the *PICTURE statement* to create a template for printing numbers.

General form, PROC FORMAT with the PICTURE statement:

```
PROC FORMAT;
    PICTURE format-name
              value-or-range='picture';
RUN;
```

where

format-name

is the name of the format you are creating.

value-or-range

is the individual value or range of values you want to label.

picture

specifies a template for formatting values of numeric variables. The template is a sequence of characters enclosed in quotation marks. The maximum length for a picture is 40 characters.

Ways to Specify Pictures

Pictures are specified with three types of characters:

- digit selectors
- message characters
- directives.

Consider using digit selectors and message characters first. You will learn about directives a little later.

Digit selectors are numeric characters (0 through 9) that define positions for numeric values. If you use *nonzero* digit selectors, zeros are added to the formatted value as needed. If you use *zeros* as digit selectors, no zeros are added to the formatted value.

In the picture definitions below you can see the difference between using nonzero digit selectors (99) and zero digit selectors (00) on the formatted values.

Picture Definition	Data Values	Formatted Values
`picture month 1-12='99';`	01	01
	1	01
	12	12
`picture month 1-12='00';`	01	1
	1	1
	12	12

Message characters are nonnumeric characters that print as specified in the picture. They are inserted into the picture after the numeric digits are formatted. Digit selectors must come before message characters in the picture definition. The prefix option can be used to append text in front of any digits. In the picture definition below, the text string *JAN* consists of message characters.

Picture Definition	Data Value	Formatted Value
`picture month 1='99 JAN';`	1	01 JAN

Example

The following PICTURE statement contains both digit selectors and message characters. Because the RAINAMT. format has nonzero digit selectors, values are printed with leading zeros. The keyword OTHER is used to print values and message characters for any values that do not fall into the specified range.

```
proc format;
   picture rainamt
           0-2='9.99 slight'
           2<-4='9.99 moderate'
           4<-<10='9.99 heavy'
           other='999 check value';
run;
data rain;
   input Amount;
   datalines;
   4
   3.9
   20
   .5
   6
   ;
run;
proc print data=rain;
   format amount rainamt.;
run;
```

The following output shows the values with the RAINAMT. format applied.

Obs	Amount
1	4.00 moderate
2	3.90 moderate
3	020 check value
4	0.50 slight
5	6.00 heavy

Next you learn about using *directives* to specify picture formats.

The final way to specify a picture is with a directive. *Directives* are special characters that you can use in the picture to format *date, time*, or *datetime* values. If you use a directive, you must specify the *DATATYPE= option* in the PICTURE statement. This option specifies that the picture applies to a SAS date, SAS time, or SAS datetime value.

General form, PICTURE statement with the DATATYPE= option:

```
PICTURE format-name
      value-or-range, 'picture' (DATATYPE=SAS-date-value-type);
```

where

format-name

is the name of the format you are creating.

value-or-range

is the individual value or range of values you want to label.

picture

specifies a template with directives for formatting values of numeric variables.

SAS-date-value-type

is either *DATE*, *TIME* or *DATETIME*.

Guidelines for Specifying Directives

The percent sign (%) followed by a letter indicates a directive. Directives that you can use to create a picture format are listed in the table below.

Directive	Result
%a	abbreviated weekday name
%A	full weekday name
%b	abbreviated month name
%B	full month name
%d	day of the month as a number 1-31, with no leading zero
%H	24-hour clock as a number 0-23, with no leading zero
%I	12-hour clock as a number 1-12, with no leading zero
%j	day of the year as a number 1-366, with no leading zero
%m	month as a number 1-12, with no leading zero
%M	minute as a decimal number 0-59, with no leading zero
%p	AM or PM
%S	second as a number 0-59, with no leading zero
%U	week number of the year (Sunday is the first day of the week) as a number 0-53, with no leading zero
%w	weekday as a number (l=Sunday, to 7)
%y	year without century as a number 0-99, with no leading zero
%Y	year with century as a number

Although directives generally return numeric values with no leading zeros, you can add 0 in the directive so that if a one-digit numeric value is returned, it is preceded bya 0.

As shon below, when you create a picture with directives, the number of characters inside quotation marks is the maximum length of the formatted value. You must add trailing blanks to the directive if your values will contain more characters than the picture. The formatted value will be truncated if you do not.

Formatted Result	Picture Definition
d d - m m m y y y y 1 2 3 4 5 6 7 8 9 10	'%0d-%b%Y ' 1 2 3 4 5 6 7 8 9 10

Example

Suppose you want to display values for employee hire dates in the format *dd-mmmyyyy* (such as *25-JAN2000*). This format requires spaces for 10 characters.

The following code creates this format. There are a few things you should notice about the picture definition:

- The keyords LOW and HIGH are used to include all values.
- The 0 in the %d directive indicates that if the day of the month is one digit, it should be preceded by a 0.
- Because there are only eight characters inside the single quotation marks, you must add two blank spaces to set the length to 10.

```
proc format;
   picture mydate
           low-high='%0d-%b%Y  ' (datatype=date);
run;

proc print data=sasuser.empdata
     (keep=division hireDate lastName obs=5);
   format hiredate mydate.;
run;
```

The output below shows the values for **`HireDate`** formatted with the MYDATE. picture format.

Obs	Division	HireDate	LastName
1	FLIGHT OPERATIONS	11-MAR199	MILLS
2	FINANCE & IT	19-DEC198	BOWER
3	HUMAN RESOURCES & FACILITIES	12-MAR198	READING
4	HUMAN RESOURCES & FACILITIES	15-OCT198	JUDD
5	AIRPORT OPERATIONS	19-DEC198	WONSILD

Tip For more information about using the PICTURE statement, see the documentation for the FORMAT procedure.

Managing Custom Formats

Using FMTLIB with PROC FORMAT to Document Formats

When you have created a large number of permanent formats, it can be easy to forget the exact spelling of a specific format name or its range of values. Remember that adding the keyword *FMTLIB* to the PROC FORMAT statement displays a list of all the formats in the specified catalog, along with descriptions of their values.

```
libname library 'c:\sas\newfmt';
proc format lib=library fmtlib;
run;
```

You can also use the *SELECT* and *EXCLUDE* statements to process specific formats rather than an entire catalog.

General form, PROC FORMAT with FMTLIB and the SELECT and EXCLUDE statements:

```
PROC FORMAT LIB=library FMTLIB;
     SELECT format-name;
     EXCLUDE format-name;
RUN;
```

where

library

is the name of the library where the formats are stored. If you do not specify the LIB= option, formats in the *Work* library are listed.

format-name

is the name of the format you want to select or exclude.

Example

The following code displays only the documentation for the $ROUTES. format. Notice that you do not use a period at the end of the format name when you specify the format in the SELECT statement.

```
libname library 'c:\sas\newfmt';
proc format lib=library fmtlib;
   select $routes;
run;
```

Table 17.1: SAS Output

FORMAT NAME : $ROUTES LENGTH: 7 NUMBER OF VALUES: 5 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 7 FUZZ: 0		
START	**END**	**LABEL (VER. V7|V8 29AUG2002:11:13:14)**
		Missing
Routel	Routel	Zone 1
Route2	Route4	Zone 2
Route5	Route7	Zone 3
OTHER	**OTHER**	Unknown

Tip If you specify more than one format on the SELECT or EXCLUDE statement, separate each format name with a space as follows:

```
select $routes newdate;
```

Using PROC CATALOG to Manage Formats

Because formats are saved as catalog entries, you can use the CATALOG procedure to manage your formats. Using PROC CATALOG, you can

- create a listing of the contents of a catalog
- copy a catalog or selected entries within a catalog
- delete or rename entries within a catalog.

General form, PROC CATALOG step:

```
PROC CATALOG CATALOG=libref.catalog;
     CONTENTS <OUT=SAS-data-set>;
     COPY OUT=libref.catalog <options>;
     SELECT entry-name.entry-type(s);
     EXCLUDE entry-name.entry-type(s);
     DELETE entry-name.entry-type(s);
RUN;
QUIT;
```

where

libref.catalog

with the CATALOG= argument is the SAS catalog to be processed.

SAS-data-set

is the name of the data set that will contain the list of the catalog contents.

libref.catalog

with the OUT= argument is the SAS catalog to which the catalog entries will be copied.

entry-name.entry-type(s)

are the full names of catalog entries (in the form *name.type*) that you want to process.

Caution If you are using the SAS Learning Edition, you will not be able to submit a PROC CATALOG step because the CATALOG procedure is not included in the software.

Example

The first PROC CATALOG step below copies the $ROUTES. format from the *Library.Formats* catalog to the *Work.Formats* catalog. Notice that in the SELECT statement, you specify the $ROUTES. character format using the full catalog entry name, ROUTES.FORMATC.

```
proc catalog catalog=library.formats;
   copy out=work.formats;
   select routes.formatc;
run;
proc catalog cat=work.formats;
   contents;
run;
quit;
```

The second PROC CATALOG step displays the contents of the *Work.Formats* catalog. A note is written to the log when the format is copied from one catalog to another.

Contents of Catalog WORK.FORMATS					
#	Name	Type	Create Date	Modified Date	Description
1	DATES	FORMAT	25Apr11:09:29:20	25Apr11:09:29:20	
2	MYDATE	FORMAT	25Apr11:09:55:18	25Apr11:09:55:18	
3	RAINAMT	FORMAT	25Apr11:09:53:47	25Apr11:09:53:47	

Tip For more information about PROC CATALOG, including other statements and options that you can use, see the SAS documentation.

Using Custom Formats

Overview

After you have created a custom format, you can use SAS statements to permanently assign the format to a variable in a DATA step, or you can temporarily specify a format in a PROC step to determine the way that the data values appear in output. You should already be familiar with referencing a format in a FORMAT statement.

Another way to assign, change, or remove the format associated with a variable in an existing SAS data set is to use the DATASETS procedure to modify the descriptor portion of a data set.

General form, DATASETS procedure with the MODIFY and FORMAT statements:

PROC DATASETS LIB=*SAS-library* <NOLIST>;

```
      MODIFY SAS-data-set;
      FORMAT variable(s)format;
QUIT;
```

where

SAS-library

is the name of the SAS library that contains the data you want to modify.

NOLIST

suppresses the directory listing.

SAS-data-set

is the name of the SAS data set you want to modify.

variable

is the name of one or more variables whose format you want to assign, change, or remove.

format

is the name of a format to apply to the variable or variables listed before it. If you do not specify a format, any format associated with the variable is removed.

Note The DATASETS procedure is interactive and will run until you issue the QUIT statement.

Example

In the following code, two variables in the SAS data set *Flights* are changed. The format $DEST. is associated with the variable **Dest** and the format is removed from the variable **Baggage**.

```
proc datasets lib=Mylib;
   modify flights;
   format dest $dest.;
   format baggage;
quit;
```

Using a Permanent Storage Location for Formats

When you permanently associate a format with a variable in a data set, it is important to ensure that the format you are referencing is stored in a permanent location. Remember that the storage location for the format is determined when the format is created in the FORMAT procedure.

When you create formats that you want to use in subsequent SAS sessions, it is useful to

1. assign the *Library* libref to a SAS library in the SAS session in which you are running the PROC FORMAT step
2. specify LIB=LIBRARY in the PROC FORMAT step that creates the format
3. include a LIBNAME statement in the program that references the format to assign the *Library* librefto the library that contains the permanent format catalog.

You can store formats in any catalog you choose. However, you must identify the format catalogs to SAS before you can access them. You learn about this in the next topic.

When a format is referenced, SAS automatically looks through the following libraries in this order:

- *Work.Formats*
- *Library.Formats*.

The *Library* librefis recommended for formats because it is automatically searched when a format is referenced. If you store

formats in libraries or catalogs other than those in the default search path, you must use the FMTSEARCH= system option to tell SAS where to find your formats.

General form, FMTSEARCH= system option:

```
OPTIONS FMTSEARCH= (catalog-1 catalog-2…catalog-n);
```

where

catalog

is the name of one or more catalogs to search. The value of *catalog* can be either *libref* or *libref.catalog*. If only the librefis given, SAS assumes that *Formats* is the catalog name.

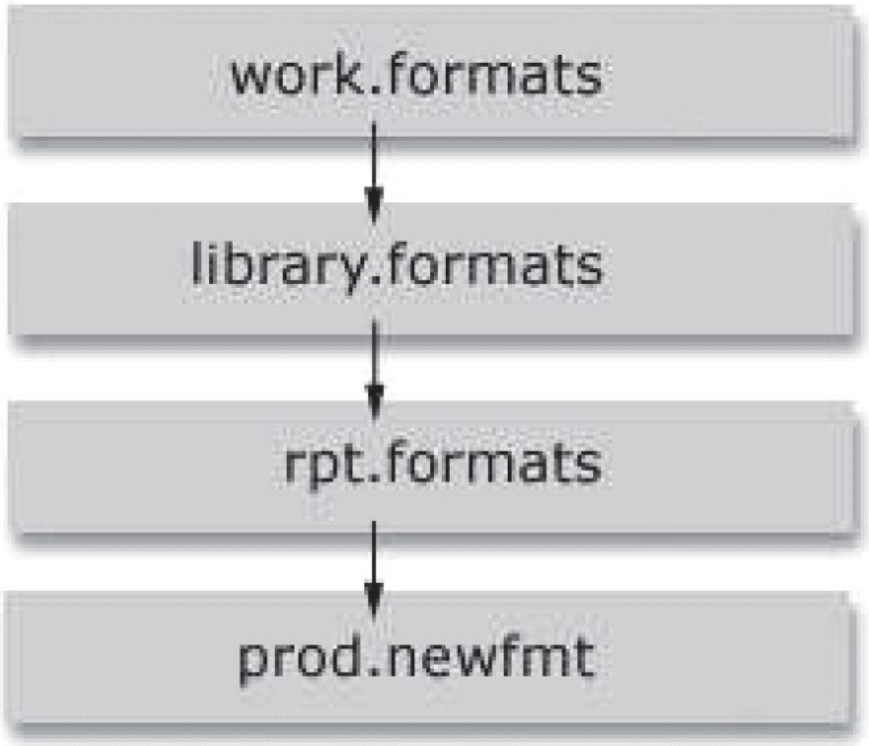
The *Work.Formats* catalog is always searched first, and the *Library.Formats* catalog is searched next, unless one or both catalogs appear in the FMTSEARCH= list.

Example

Suppose you have formats stored in the *Rpt* library and in the *Prod.Newfmt* catalog. The following OPTIONS statement tells SAS where to find your formats:

```
options fmtsearch=(rpt prod.newfmt);
```

Because no catalog is specified with the *Rpt* libref, the default catalog name *Formats* is assumed. This OPTIONS statement creates the following search order:



Because the *Work* and *Library* librefs were not specified in the FMTSEARCH= option, they are searched in default order.

Substituting Formats to Avoid Errors

Consider what happens if you forget to specify a catalog in the FMTSEARCH= option, misspell a format name, or make some other mistake that causes SAS to fail to locate the format you have specified.

By default, the *FMTERR* system option is in effect. If you use a format that SAS cannot load, SAS issues an error message and stops processing the step. To prevent this, you must change the system option FMTERR to NOFMTERR. When *NOFMTERR* is in effect, SAS substitutes a format for the missing format and continues processing.

General form, FMTERR system option:

```
OPTIONS FMTERR | NOFMTERR;
```

where

FMTERR

specifies that when SAS cannot find a specified variable format, it generates an error message and stops processing. Substitution does not occur.

NOFMTERR

replaces missing formats with the *w*. or $*w*. default format and continues processing.

Example

Suppose the FMTERR system option is in effect. In a previous example, we created the $ROUTES. format to group airline routes into zones. In the following code, the $ROUTES. format is misspelled:

```
proc print data=sasuser.cargorev(obs=10);
   format route $route.;
run;
```

Because FMTERR is in effect, the format cannot be located and SAS stops processing. An error message is written to the log.

Table 17.2: SAS Log

```
30   proc print data=sasuser.cargorev(obs=10);
31      format route $route.;
ERROR: The format $ROUTE was not found or could not be loaded.
32   run;


NOTE: The SAS System stopped processing this step because of errors.
```

If the NOFMTERR system option is specified, substitution occurs and SAS continues to process the step.

```
options nofmterr;
proc print data=sasuser.cargorev(obs=10);
   format Route $route.;
run;
```

SAS substitutes the $w format for the $ROUTE. format that it could not locate. No message is written to the log and processing continues. You can see from the output that the format you intended to use has not been applied.

Obs	Month	Date	RevCargo	Route
1	1	14610	2260	Route2
2	1	14610	220293	Route3
3	1	14610	4655	Route1
4	1	14610	4004	Route1
5	1	14611	8911	Route1
6	1	14611	102900	Route3
7	1	14612	1963	Route3
8	1	14612	3321	Route5

9	1	14612	2562	Route3
10	1	14612	9447	Route1

Creating Formats from SAS Data Sets

Overview

You have seen that you can create a format by specifying values and labels in a PROC FORMAT step. You can also create a format from a SAS data set that contains value information (called a control data set). To do this, you use the *CNTLIN= option* to read the data and create the format.

General form, CNTLIN= option in PROC FORMAT:

```
PROC FORMAT LIBRARY=libref.catalog
            CNTLIN=SAS-data-set;
```

where

libref.catalog

is the name of the catalog in which you want to store the format.

SAS-data-set

is the name of the SAS data set that you want to use to create the format.

Example

Suppose you have a SAS data set named *Routes* that has the variables required to create a format. You specify the data set in the CNTLIN= option as follows:

```
proc format lib=library.formats cntlin=sasuser.routes;
run;
```

As you can see, the code for creating a format from a SAS data set is simple. However, the control data set must contain certain variables before it can be used to create a format, and most data sets must be restructured before they can be used.

Rules for Control Data Sets

When you create a format using programming statements, you specify the name of the format, the range or value, and the label for each range or value as in the VALUE statement below:

```
value rainfall 0='none';
```

The control data set you use to create a format must contain variables that supply this same information. That is, the data set specified in the CNTLIN= option

- must contain the variables **`FmtName, Start`**, and **`Label`**, which contain the format name, value or beginning value in the range, and label.
- must contain the variable **`End`** if arange is specified. If there is no **`End`** variable, SAS assumes that the ending value of the format range is equal to the value of **`Start`**.
- must contain the variable **`Type`** for character formats, unless the value for **`FmtName`** begins with a $.
- must be sorted by **`FmtName`** if multiple formats are specified.

Now consider how you create a correctly structured data set.

Example

Overview

Suppose you want to create a format that labels a three-letter airport code with the name of the city where the airport is located. You have a data set, *Sasuser.Acities*, that contains airport codes and airport cities. However, the data does not have the required variables for the CNTLIN= option.

Table 17.3: SAS Data Set Sasuser.Acities (Partial Listing)

City Where Airport is Located	Start Point	Airport Name	Country Where Airport is Located
Auckland	AKL	International	New Zealand
Amsterdam	AMS	Schiphol	Netherlands
Anchorage, AK	ANC	Anchorage International Airport	USA
Stockholm	ARN	Arlanda	Sweden
Athens (Athinai)	ATH	Hellinikon International Airport	Greece
Birmingham, AL	BHM	Birmingham International Airport	USA
Bangkok	BKK	Don Muang International Airport	Thailand

To create a format from this data set, you need to

1. List data set variables
2. Restructure the data

Step 1: List Data Set Variables

Remember that you need to have the variables **FmtName, Start**, and **Label**. You can submit a PROC CONTENTS step to get a listing of the variables in the *Sasuser.Acities* data set.

Partial Output

```
proc contents data=sasuser.acities;
run;
```

Alphabetic List of Variables and Attributes				
#	Variable	Type	Len	Label
1	City	Char	22	City Where Airport is Located
2	Code	Char	3	Start Point
4	Country	Char	40	Country Where Airport is Located
3	Name	Char	50	Airport Name

Tip You can also get a list of variable names by using PROC DATASETS with a CONTENTS statement or by viewing the properties of the SAS data set in the SAS Explorer window.

Step 2: Restructure the Data

Once you have looked at the data and kno the variable names, you are ready to write a DATA step to manipulate the data. The variable **Code** is the three-letter airport code and the variable **City** is the city where the airport is located. You can rename the variable **Code** to **Start** and the variable **City** to **Label**, but you also need to create the variable **FmtName**.

The code below is an efficient way to get your data ready to use. The DATA step uses

- the KEEP statement to write only the specified variables to the output data set
- the RETAIN statement to create the variable **FmtName** and set the value to '$airport'
- the RENAME statement to rename the variable **Code** to **Start** (you do not need a variable named **End** because you are labeling discrete values rather than ranges) and to rename the variable **City** to **Label**.

```
data sasuser.aports;
   keep Start Label FmtName;
   retain FmtName '$airport';
   set sasuser.acities (rename=(Code=Start
       City= Label));
run;

proc print data=sasuser.aports(obs=10) noobs;
run;
```

Below is the listing of the first ten observations in the new data set *Sasuser.Aports*.

FmtHame	Label	Start
$airport	Auckland	AKL
$airport	Amsterdam	AMS
$airport	Anchorage, AK	ANC
$airport	Stockholm	ARN
$airport	Athens (Athinai)	ATH
$airport	Birmingham. AL	BHM
$airport	Bangkok	BKK
$airport	Nashville, TN	BNA
$airport	Boston, MA	BOS
$airport	Brussels (Bruxelles)	BRU

This data set is now in the proper format to be used to create a format using the CNTLIN= option. Next consider the code that creates the format from this data set. Once you have the data in the proper format, you can use the CNTLIN= option to create the format. The first PROC FORMAT step creates a format from the data set *Sasuser.Aports*. The second PROC FORMAT step documents the new format.

```
proc format library=sasuser cntlin=sasuser.aports;
run;

proc format library=sasuser fmtlib;
   select $airport;
run;
```

The first few lines of the output are shown below.

Table 17.4: Partial SAS Output

FORMAT NAME : $AIRPORT LENGTH: 22 NUMBER OF VALUES: 52 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 22 FUZZ: 0

START	END	LABEL (VER. V7|V8 21OCT2002:14:13:14)
AKL	AKL	Auckland
AMS	AMS	Amsterdam
ANC	ANC	Anchorage, AK
ARN	ARN	Stockholm
ATH	ATH	Athens (Athinai)
BHM	BHM	Birmingham, AL
BKK	BKK	Bangkok

Apply the Format

Consider the format that is applied to the data set *Sasuser.Cargo99*. The following PROC PRINT code assigns the $AIRPORT. format to both the **`Dest`** and **`Origin`** variables:

```
proc print data=sasuser.cargo99 (obs=5);
   var origin dest cargorev;
   format origin dest $airport.;
run;
```

Obs	Origin	Dest	CargoRev
1	RDU	LHR	$111,720.00
2	RDU	LHR	$109,270.00
3	RDU	LHR	$109,270.00
4	RDU	LHR	$116.130.00
5	RDU	LHR	$108,290.00

Tip For more information about using the CNTLIN= option, see the SAS documentation for the FORMAT procedure.

Creating SAS Data Sets from Custom Formats

Overview

You know how to create a format from a SAS data set, but what if you want to create a SAS data set from a format? To do this, you use the CNTLOUT= option.

General form, CNTLOUT= option in PROC FORMAT:

```
PROC FORMAT LIBRARY=libref.catalog CNTLOUT=SAS-data-set;
      SELECT format-nameformat-name...;
      EXCLUDE format-nameformat-name...;
RUN;
```

where

libref.catalog

is the name of the catalog in which the format is located.

SAS-data-set

is the name of the SAS data set that you want to create.

format-name

is the name of the format that you want to select or exclude.

The output control data set will contain variables that completely describe all aspects of each format, including optional settings. The output data set contains one observation per range per format in the specified catalog. You can use either the *SELECT* or *EXCLUDE* statement to include specific formats in the data set.

Creating a SAS data set from a format is very useful when you need to add information to a format but no longer have the SAS data set you used to create the format. When you need to update a format, you can

1. create a SAS data set from the values in a format using CNTLOUT=
2. edit the data set using any number of methods
3. create a format from the updated SAS data set using CNTLIN=.

Next consider these steps individually.

Example

Overview

In the last example, you created the $AIRPORT. format. Suppose you want to add the following airport codes to the format:

Value	Label
YYC	Calgary, AB
YYZ	Toronto, ON
YQB	Quebec, QC
YUL	Montreal, QC

Step 1: Create a SAS Data Set from the Format

First, you write the $AIRPORT. format out as a SAS data set. In the code below, the output data set is named *Sasuser.Fmtdata*. The SELECT statement is used so that the resulting data set has only the data for the $AIRPORT. format. Without the SELECT statement, the data would have values for all the formats in the *Sasuser.Formats* catalog.

```
proc format lib=sasuser cntlout=sasuser.fmtdata;
   select $airport;
run;
```

When you use the CNTLOUT= option, SAS creates an output data set that has many variables for storing information about the format. The output data set *Sasuser.Fmtdata* has 50 rows and 21 columns. In the PRINT procedure below, the VAR statement specifies only a few of the variables to print:

```
proc print data=sasuser.fmtdata (obs=5) noobs;
   var fmtname start end label min max
       default length fuzz;
run;
```

FMTNAME	START	END	LABEL	MIN	MAX	DEFAULT	LENGTH	FUZZ
AIRPORT	AKL	AKL	Auckland	1	40	22	22	0
AIRPORT	AMS	AMS	Amsterdam	1	40	22	22	0
AIRPORT	ANC	ANC	Anchorage, AK	1	40	22	22	0
AIRPORT	ARN	ARN	Stockholm	1	40	22	22	0
AIRPORT	ATH	ATH	Athens (Athinai)	1	40	22	22	0

As you can see, the data set contains **End** and other variables that were not in the original data. When you use the CNTLIN= option, if there is no **End** variable in the data set, SAS assumes that the **Start** and **End** variables have the same value. When you write the format out as a data set using the CNTLOUT= option, both variables are in the data set.

Step 2: Edit the Data Set

The next step in updating the format is to edit the data set. You could use PROC SQL or a DATA step to add lines to the data set, or you could add rows using VIETABLE. Whatever method you choose, you *must* add values for the **FmtName, Start, End**, and **Label** variables. If the values for **Start** and **End** are the same, you must enter values for both variables or SAS will return an error. You do not have to add values for the other variables in the data set.

Step 3: Create a Format from the SAS Data Set

Once the data set is edited and saved, you can create a format from the data set using the CNTLIN= option. The following code creates the $AIRPORT. format and then uses FMTLIB to document it:

```
proc format library=sasuser cntlin=sasuser.fmtdata;
run;

proc format lib=sasuser fmtlib;
   select $airport;
run;
```

The partial output shown below includes the new values in the format.

Table 17.5: Partial SAS Output

FORMAT NAME : $AIRPORT LENGTH: 22 NUMBER OF VALUES: 56 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH: 22 FUZZ: 0

START	END	LABEL (CONT'D)
SYD	END	Sydney, New South Wales
VIE	VIE	Wien (Vienna)
WLG	WLG	Wellington
YQB	YQB	Quebec, QC
YUL	YUL	Montreal, QC
YYC	YYC	Calgary, AB
YYZ	YYZ	Toronto, ON

Tip For more information about using the CNTLOUT= option, see the SAS documentation for the FORMAT procedure.

Summary

Contents

This section contains the following topics.

Text Summary

Creating Custom Formats Using the VALUE Statement

Character and numeric formats are created by using VALUE statements in a FORMAT procedure. When you specify a librefon the LIBRARY= option, the format is stored in the specified library. If no catalog name is specified, the format is saved in the *Formats* catalog by default.

Creating Formats with Overlapping Ranges

Use the MULTILABEL option to create a format that has overlapping ranges. When a format has overlapping ranges, the values in the format may have more than one label. This format can be used in procedures that support the MLF option.

Creating Custom Formats Using the PICTURE Statement

The PICTURE statement is used to create a template for formatting values of numeric variables. Pictures are specified using digit selectors, message characters, and directives.

Documenting Formats

Use the FMTLIB keyword in the PROC FORMAT statement to get documentation about the formats in the specified catalog. The output displays the format name, start and end values, and the label. You can also use the SELECT and EXCLUDE statements to process specific formats rather than an entire catalog.

Managing Formats

Because formats are saved as catalog entries, you use PROC CATALOG to copy, rename, delete, or create a listing of the entries in a catalog.

Using Custom Formats

Once you have created a format, you can reference it as you would reference a SAS format. If you have stored the format in a location other than *Work.Formats*, you must use the FMTSEARCH= system option to add the location to the search path so that SAS can locate the format. It can be useful to change the default FMTERR system option to NOFMTERR so

that if SAS does not find a format you reference, it will substitute the *w*. or $*w*. format and continue processing.

You can permanently associate a format with a variable by modifying the data set using PROC DATASETS.

Creating Formats from SAS Data Sets

Use the CNTLIN= option to specify a SAS data set that you want to use to create a format. The SAS data set must contain the variables **FmtName, Start**, and **Label**. If the values have ranges, there must also be an **End** variable.

Creating SAS Data Sets from Formats

Use the CNTLOUT= option to create a SAS data set from a format. This is useful for maintaining formats because you can easily update a SAS data set.

Syntax

```
OPTIONS FMTSEARCH=(catalog catalog);
OPTIONS FMTERR | NOFMTERR;
LIBNAME libref 'SAS-data-library';
PROC FORMAT LIBRARY=libref;
     VALUE format-name (MULTILABEL)
             rangel = 'labell'
             rangen='labeln';
RUN;
PROC FORMAT;
     PICTURE format-name
          rangel = 'picture1'
          rangen='picturen';
RUN;
PROC FORMAT;
     PICTURE format-name
             value-or-range='picture'
             (DATATYPE=SAS-date-value-type);
RUN;
PROC FORMAT LIB=library FMTLIB;
     SELECT format-name1 format-name2;
     EXCLUDE format-name1 format-name2;
RUN;
PROC FORMAT LIBRARY=libref.catalog
     CNTLOUT=SAS-data-set;
     SELECT format-name1 format-name2;
     EXCLUDE format-name1 format-name2;
RUN;
PROC FORMAT LIBRARY=libref.catalog
     CNTLOUT=SAS-data-set;
     SELECT format-name1 format-name2;
     EXCLUDE format-name1 format-name2;
RUN;
PROC CATALOG CATALOG=libref.catalog;
     CONTENTS <OUT=SAS-data-set>;
     COPY OUT=libref.catalog<options>;
     SELECT entry-name.entry-type(s);
     EXCLUDE entry-name.entry-type(s);
     DELETE entry-name.entry-type(s);
RUN;
QUIT;
PROC DATASETS LIB=SAS-library;
     MODIFY SAS-data-set;
     FORMAT variable(s) format;
QUIT;
```

Sample Programs

Creating a Multilabel Format

```
proc format;
   value dates multilabel)
```

```
          '01jan2000'd - '31mar2000'd = '1st Quarter'
          '01apr2000'd - '30jun2000'd = '2nd Quarter'
          '01jul2000'd - '30sep2000'd = '3rd Quarter'
          '01oct2000'd - '31dec2000'd = '4th Quarter'
          '01jan2000'd - '30jun2000'd = 'First Half of Year'
          '01jul2000'd - '31dec2000'd = 'Second Half of Year';
run;
```

Creating a Picture Format

```
proc format;
   picture rainamt
           0-2='9.99 slight'
           2<-4='9.99 moderate'
           4<-<10='9.99 heavy'
           other='999 check value';
run;
```

Creating a Picture Format Using Directives

```
proc format;
   picture mydate
           low-high='%0d-%b%Y ' (datatype=date);
run;
```

Restructuring a SAS Data Set and Creating a Format from the Data

```
data sasuser.aports;
   keep Start Label FmtName;
   retain FmtName '$airport';
   set sasuser.acities (rename=(Code=Start
       City= Label));
run;

proc format library=sasuser cntlin=sasuser.aports;
run;
```

Creating a SAS Data Set from a Format

```
proc format lib=sasuser cntlout=sasuser.fmtdata;
   select $airport;
run;
```

Points to Remember

- By default, SAS searches for formats in the *Work.Formats* and *Library.Formats* catalogs. If you store formats in other catalogs, you must you use the FMTSEARCH= system option to tell SAS where to look for your formats.
- You can use the CNTLIN= option to create a format from a SAS data set, but the data must contain the following variables:
  - **`FmtName, Start`**, and **`Label`**
  - **`Type`** for character formats, unless the value for **`FmtName`** begins with a $
  - **`End`** if arange is specified

Quiz

Select the best answer for each question. After completing the quiz, check your answers using the answer key in the appendix.

1. Which SAS system option is used to identify format catalogs to SAS? ?

a. FMTERR

b. FMTLIB

c. NOFMTERR

d. FMTSEARCH=

2. Given the following PROC FORMAT step, how is the value *70* displayed when the AGEGRP. format is applied? ?

```
proc format;
   picture agegrp
           1-<13='00 Youth'
           13-<20='00 Teen'
           20-<70='00 Adult'
           70-high='000 Senior';
run;
```

a. 000 Senior

b. 70 Adult

c. 70 Senior

d. 070 Senior

3. When the NOFMTERR system option is in effect, what happens when SAS encounters a format it cannot locate? ?

a. Creates the format in the default *Work.Formats* directory and continues processing.

b. Substitutes the $*w*. or *w*. format and continues processing.

c. Stops processing and writes an error message to the log.

d. Skips processing at that step and continues with the next step and writes a note to the log.

4. Which of the following variables must be in the data set that is specified on the CNTLIN= option? ?

a. `End`

b. `FmtName`

c. `Value`

d. `Description`

5. Given the following code, what option is missing? ?

```
proc format;
   value times ?)
         '00:00't-'04:59't = 'Red Eye'
         '05:00't-'11:59't = 'Morning'
         '12:00't-'17:59't = 'Afternoon'
         '18:00't-'23:59't = 'Evening'
         '00:00't-'11:59't = 'AM'
         '12:00't-'23:59't = 'PM';
run;
```

a. MULTILABEL

b. MULTIRANGE

c. MLF

d. MULTIFORMAT

6. Which PROC FORMAT option is used to create a SAS data set from a format? ?

a. CNTLIN=

b. LIB=

c. CNTLOUT=

d. FMTLIB

7. Given the following OPTIONS statement, in what order will SAS search to find a user-defined format? ?

```
options fmtsearch=(work abc.newfmt sasuser);
```

a. **Work.Formats ⇨ Abc.Newfmt ⇨ Sasuser.Formats ⇨ Library.Formats**

b. **Work.Formats ⇨ Library.Formats ⇨ Abc.Newfmt ⇨ Sasuser.Formats**

c. **Work.Formats ⇨ Abc.Newfmt ⇨ Sasuser.Format**

d. the default search order

8. What option is used with PROC FORMAT to document the formats in a particular format catalog? ?

a. FMTSEARCH

b. FMTERR

c. CATALOG

d. FMTLIB

9. Which set of statements would you add to the PROC CATALOG code to copy the LEVELS, and $PICKS. formats from the *Sasuser.Formats* catalog to the *Work.Formats* catalog? ?

```
proc catalog cat=sasuser.formats;
   ?
   ?
run;
```

a.
```
a. copy out=sasuser.formats;
     select levels.format $picks.format;
```

b.
```
b. copy out=work.formats;
     select levels $picks;
```

c.
```
c. copy out=work.formats;
     select levels.format picks.formatc;
```

d.
```
d. copy out=work.formats;
     select levels.format $picks.format;
```

10. Given the following PROC FORMAT step, how is the value *6.1* displayed when the SKICOND format is applied? ?

```
proc format;
   value skicond
          0-3='Poor'
         3<-6='Fair'
         6<-9='Good'
         9<-high='Excellent';
run;
```

a. 6.1

b. Fair

c. Good

d. .

Answers

1. Correct answer: d

By default, SAS searches for custom formats in the *Work* and *Library* libraries. The FVITSEARCH ~ system option specifies other catalogs to search when a format is referenced.

2. Correct answer: c

A non-inclusive range is used such that the age at the high end of the range is not included. To create the picture format, three zeros are used to create a position for a three-digit numeric value. Because zero is used as a digit selector rather than a nonzero value, leading zeros are not included in the formatted value.

3. Correct answer: b

By default, FMTERR is in effect and SAS stops processing if it cannot find a format that is referenced. When NOFMTERR is in effect, SAS substitutes the $*w*. or *w*. format and continues processing.

4. Correct answer: b

A data set that is used to create a format with the CNTEIN option must have the variables **`FmtName, Start`**, and **`Label`**. If a range is specified, it must also include the variable **`End`**.

5. Correct answer: a

The format created by this value statement has overlapping ranges, so the MULTILABEL option must be used. A multilabel format can be used by any procedure that supports the MLF option.

6. Correct answer: c

The CNTLOUT= option is used to create a SAS data set from a format.

7. Correct answer: b

SAS will search in the order specified on the FMTSEARCH= option. By default, SAS searches in the *Work* and *Library* libraries first unless they are specified on the option. Because *Library* is not specified here, it is searched after *Work*.

8. Correct answer: d

The FMTLIB keyword is used to document the formats in a catalog. You can use the SELECT and EXCLUDE statements to process specific formats rather than the entire catalog.

9. Correct answer: c

In the COPY statement, OUT= specifies the catalog to which you want to copy the format catalog entry. In the SELECT statement you specify the catalog entries by their entire name. Remember that numeric formats are stored with the extension *.FORMAT* and character formats are stored with the extension *.FORMATC*.

10. Correct answer: c

The value *6.1* falls in the range 6<-9, which is labeled 'Good.' The non-inclusive range docs not include the value *6*, but it docs include everything above *6*.